دليلك نحو بناء نظام MVC بسيط



شبكة ديوبي الإلكترونية

هذا الكتيب هو الإصدار الثاني لكتيب آخر أطلق منذ فترة على مدونة ديوبي المجانية على وورد بريس ، أما بعد إنشاء شبكة ديوبي الإلكترونية فقد تم إطلاق الإصدار الثاني تحت اسم الشبكة .

تم كتابة هذا الكتيب من قبل : محمد ديوب .

ما هو MVC ؟

هو نظام مستخدم في جميع أطر العمل Frameworks و هو إختصار للتالي :

(**M**odel – **V**iew – **C**ontroller)

ما هو إطار العمل framework ؟

Framework ، أو إطار العمل وهو عبارة عن تطبيقات متكاملة تساعد المبرمج على أداء أعماله البرمجية بسرعة أكبر و جهد أقل مع توفر الإتقان والإحترافية في العمل .

ماذا أحتاج لكي أبني هذا النظام ؟

يجب أن تتوفر لديك نسخة php5 و أنصحك باستخدام سيرفر **wamp** (تستطيع أن تحمله من الموقع الرسمي مجاناً) إذا كنت تستخدم ويندوز أما في حال كنت تستخدم لينكس فالبطبع أنت تعمل على lamp أو قمت بتنصيب كل سيرفر على حدى بشكل يدوي ! .

> هذا الكتيب لن يعملك البرمجة بلغة php لذلك يجب أن يكون لك خبرة سابقة في التعامل مع اللغة أو أن تعرف على الأقل اساسيات برمجة الكائنات و الوراثة .

حسناً لننطلق...

قم بإنشاء مجلد جديد و سميه كما تشاء (أنا سوف اسميه **diouby**) و أنشىء فيه المجلدات التالية

1. **classes** : نضع فيه الكائنات التي سوف يتم تضمينها في النظام .
2. **controllers** : هذا المجلد سوف يجوي عنصر التحكم .
3. **include** : في هذا المجلد يتم تضمين الملفات الأساسية للنظام .

حسناً .. بعد أن ننشئ المجلدات التالية ننشئ في المجلد الرئيسي **diouby** ملف **index.php** و نكتب فيه الكود التالي :

```
<head>
 <title>نظام MVC</title>
<meta http-equiv="Content-Language" content="ar-sy">
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
```

```
<?php
//تضمين الملف الأساسي
include_once ("include/start.php");
?>
```

حسناً بما أنك مبرمج ويب لديك خلفية لا بأس بها يفترض بك أن تعرف ما هو الكود المكتوب في الأعلى بين وسمي **head** و إن كنت لا تعرف فهو كود يجعل الصفحة بترميز **utf-8** لكي تتوافق مع اللغة العربية .

يجب عليك تضمين الكود الخاص بالترميز في كل الصفحات التي سننشئها لاحقاً لأنني لن أشير عليه فيما بعد .

أما في كود **php** فإنك تلاحظ أننا قمنا بتضمين ملف أسميناه الملف الأساسي .

ما هو الملف الأساسي ؟

هو ملف نقوم بتضمينه في بداية تشغيل النظام لتحميل الخدمات الأساسية و تعيين الخيارات الخاصة بالنظام مثلاً تحديد مستوى الأخطاء و التأكد من أن إصدار **php** هو الإصدار الخامس فالنظام لا يعمل إلا على الإصدار الخامس و عدة أمور أخرى .

حسناً للنشء الآن ملف **start.php** و الموجود كما هو واضح من الكود ضمن مجلد **include** .

اكتب الكود التالي :

```
<?php
// تحديد مستوى الأخطاء
error_reporting (E_ALL);
// التأكد من إصدار php
$php_version_error=' <b><font face=tahoma color=990000 size=3>أنت بحاجة إلى
php الإصدار الخامس من </font></b>' ;
if (version_compare(phpversion(), '5', '<') == true) { die ($php_version_error); }
// ثابت يحوي اختصار للسلاش
define ('SH', '/');
// استخراج المسار الحالي
$url = realpath(dirname(__FILE__) . SH . '..' . SH) . SH;
// تخزين المسار في ثابت
define ('URL', $url);
?>
```

في البداية قمنا بتحديد مستوى الأخطاء على إظهار جميع الأخطاء ... يستفاد من هذه الخاصية في منع ظهور أخطاء معينة مثل أخطاء الاتصال بقاعدة البيانات مثلاً لديك ملف في موقعك يتصل بقاعدة البيانات و لسبب ما توقف سيرفر قاعدة البيانات فجأة في هذه الحالة سيظهر للزائر الخطأ عندما يدخل إلى موقعك أما إذا كنت قد حددت مستوى الأخطاء على منع إظهار الأخطاء الخاصة بقواعد البيانات سيظهر للزائر صفحة بيضاء بدون ظهور الخطأ و طبعاً هذا يفيد من الناحية الشكلية إضافة إلى الناحية الأمنية كما في بعض الأخطاء ....

لنعد الآن إلى درسنا , في السطر الثاني نتأكد من أن إصدار **php** هو **php5** أو إصدار أحدث و في حال كان غير ذلك نظهر رسالة الخطأ .

أما بالنسبة لثابت الفاصلة سلاش فهو لاختصار كتابة المسار فبدل أن نكتب "/" نكتب **SH**

في السطر التالي نستخرج مسار الملف و لاحظ استخدام ثابت الفاصلة في هذا السطر و في السطر التالي نضع مسار الملف في ثابت **url** لاستخدامه عند الحاجة

الآن اكتب في نهاية ملف **start.php** الكود التالي :

```
function __autoload($class_name) {
$filename = strtolower($class_name) . '.php';
$file = URL . 'classes' . SH . $filename;
 if (file_exists($file) == false) {
return false;
}
include ($file);
}
```

حسناً ما هذا الكود ؟

نحن في هذا النظام سوف نستخدم العديد من الكائنات (الكلاسات) لأعمال مختلفة

حسناً في هذه الحالة بدلاً من أن نضمن ملف الكلاس لكل كلاس نريد أن نستخدمه نستخدم الدالة **AutoLoad** و هي إحدى الدوال الجديدة في **php5**

ما هو عمل الدالة AutoLoad ؟

لنفرض أننا قمنا بتعريف كائن جديد كالتالي :

```
$gg = new gg;
```

حسناً الكائن **registry** غير موجود فيقوم النظام تلقائياً باستدعاء الدالة **AutoLoad** و تمرير اسم الكائن لها , تضيف الدالة اللاحقة **php.** إلى اسم الكائن (" لاحظ السطر الأول من الدالة ") بحيث يصبح اسم الملف **registry.php** في السطر الثاني تقوم الدالة بالبحث في مجلد **classes** عن الملف **registry.php** (" لاحظ استخدام ثابت المسار **URL** في السطر الثاني ") السطور المتبقية من الدالة تتحقق من وجود الملف فإن كان موجوداً تضمن الملف و إن كان غير موجود تعيد قيمة فارغة، دالة جميلة أليس كذلك !

إلى هنا نكون قد انتهينا من الملف **start.php**

سنقوم الآن بإنشاء كائن تسجيل يتم تضمينه داخل الكائنات الأخرى لنقل بيانات النظام

أنشء الملف **regisrty.php** داخل المجلد **classes** و اكتب بداخله الكود التالي :

```
<?php
class Registry {
private $vars = array();
}


function set($key, $var) {
if (isset($this->vars[$key]) == true) {
$error='عفواً ... هذا المفاتح موجود مسبقاً';
die($error);
}
$this->vars[$key] = $var;
return true;
}
function get($key) {
if (isset($this->vars[$key]) == false) {
return null;
}
return $this->vars[$key];
}
function remove($var) {
unset($this->vars[$key]);
}
?>
```

في البداية قمنا بإنشاء الكائن و تحديد خاصية **vars** على شكل مصفوفة نقوم بتخزين البيانات داخل هذه المصفوفة و طبعاً كأي كائن تسجيل يجب أن يحتوي على دالة للإضافة و الحذف و الاستخراج

الدلة الأولى **set** هي لإضافة (تخزين) البيانات داخل الكائن و تحتوي على بارميتران هما , **key** **vars** في السطر الأول تتحقق من ان المفتاح **key** غير مخزن سابقاً , في حال كان غير مخزن تقوم بتخزين المعلومات الجديدة و في حال كان موجود سابقاً تظهر رسالة خطأ .

الدالة الثانية **get** هي لإستخراج المعلومات المخزنة حسب المفتاح و تحتوي على بارميتر واحد هو **key** , في السطر الاول نتأكد من ان المفتاح **key** موجود في حال كان موجود تستخرج المعلومات و في حال كان غير موجود تعيد قيمة فارغة

الدالة الثالثة **remove** هي لحذف البيانات المخزنة و تحتوي على بارميتر واحد ايضاً هو **key** و عملها واضح لا داعي للشرح

حسناً الآن انتهينا من كائن التسجيل و طبعاً بما أن كائن التسجيل كما أشرنا سابقاً يتم تضمينه داخل الكائنات الأخرى لنقل بيانات النظام سنقوم بتعريفه ضمن ملف **start.php** ليعمل عند بدء عمل النظام

اكتب هذا الكود في نهاية ملف **start.php** :

```
$registry = new Registry;
```

ما هو عمل كائن المسار ؟

يقوم الكائن باستقبال الطلب (المسار) من مستخدم الموقع , ثم يحلله و يقوم بتحميل صفحة العرض المناسبة

هذا هو محتوى ملف **router.php** الموجود ضمن مجلد **classes**

```
<?php
Class Router {
private $registry;
private $path;
function __construct($registry)
```

```
{
$this->registry = $registry;
}
function setPath($path)
{
$path = trim($path, '/\\');
$path .= SH;
if (is_dir($path) == false) {
$perror= ' عفواً ... مجلد عناصر التحكم غير موجود '.$path;
die($perror);
}
$this->path = $path;
}
function loading() {
$this->getController($file, $controller, $action);
if (is_readable($file) == false) {
$this->error404('No - File');
}
include ($file);
$class = 'Controller_' . $controller;
$controller = new $class($this->registry);
if (is_callable(array($controller, $action)) == false) {
$this->error404('No - Action');
}
$controller->$action();
}

private function getController(&$file, &$controller, &$action) {
$route = (empty($_GET['route'])) ? '' : $_GET['route'];
if (empty($route)) { $route = 'index'; }
$route = trim($route, '/\\');
$parts = explode('/', $route);
$cmd_path = $this->path;
```

```
foreach ($parts as $part) {
$fullpath = $cmd_path . $part;
if (is_dir($fullpath)) {
$cmd_path .= $part . SH;
$cmd_path .= $part . SH;
array_shift($parts);
continue;
}
if (is_file($fullpath . '.php')) {
$controller = $part;
array_shift($parts);
break;
}
}
if (empty($controller)) { $controller = 'index'; };
$action = array_shift($parts);
if (empty($action)) { $action = 'index'; }
$file = $cmd_path . $controller . '.php';
}
}
?>
```

الآن إلى الشرح ...

في البداية قمنا بإنشاء خاصية التسجيل التي ستحوي كائن التسجيل الذي انشأناه سابقاً , السطر التالي فيه خاصية المسار التي ستحتوي على المسار الكامل , الدالة الموجودة تحت خاصية المسار هي المشيد و تأخذ كائن التسجيل كمعطى لها و تقوم بتعيينه لخاصية التسجيل

لمعرفة المزيد عن المشيد اذهب إلى http://www.php.net/

الدالة **setPath** التي تقوم بتحديد المجلد الذي يحوي جميع صفحات العرض في البداية تأخذا الدالة البارميتر **path** كمعطى لها تقوم بحذف العلامتين "/" و "\" من المعطى (طبعاً المعطى يقصد به اسم المجلد الذي يحتوي على صفحات العرض و سوف تفهم كيف نمرره للدالة بعد قليل) , بعد ذلك تتأكد الدالة من أن المجلد موجود فإن كان موجود تخزنه في خاصة المسار و إن لم يكن موجود تظهر رسالة خطأ .

الآن نأتي إلى شرح الدالة **getController** في البداية نستقبل المتغير **router** و نتأكد من أنه غير فارغ ثم نقسمه إلى أجزاء بوساطة الدالة **explode** التي تحوله إلى مصفوفة , نستخدم بعدها حلقة تكرار تستخرج كل عنصر من المصفوفة على حدى و تتأكد من كونه مجلد فإذا كان مجلد تقوم بإضافته إلى مسار صفحة العرض (لاحظ إستخدام الثابت **SH**) و إذا كان ملف نخرج من المصفوفة و نتأكد من أن الملف موجود فعلاً و إذا لم يكن موجود نضع القيمة الافتراضية **index** .

بعد أن انتهينا من الدالة **getController** و خزنا مسار صفحة العرض علينا تحميلها و طبعاً هذا عمل الدالة **loading**

في السطر الأول نقوم باستدعاء دالة الحصول على مسار صفحة العرض **getController** ثم نتأكد من ملف صفحة العرض موجود بالفعل فإذا كان موجود نقوم بتضمينه **include** وإذا لم يكن نخرج رسالة خطا ثم نقوم بتعريف الكلاس بسابقة **Controller** ثم اسم الملف ثم نقوم بعمل نسخة من الكلاس لاستخدامها في إخراج الوسيلة المطلوبة وبعد ذلك نقوم بالتأكد من أن الوسيلة المطلوبة هي دالة يمكن تنفيذها داخل كائن عن طريق إدخالهم على شكل مصفوفة ثم تنفيذها وإذا لم تكن نخرج رسالة خطا .

انتهينا الآن من ملف **router.php** و لكي نستخدمه اكتب التالي في الصفحة **index.php**

```
//تحميل كائن المسار
$router= new Router($registry);
$registry['router'] = $router ;

// مسار مجلد صفحات العرض
$router->setPath(URL.'controllers');

$router->loading ();
```

في البداية عملنا نسخة من كائن المسار ثم قمنا بتسجيله ليكون متغير عام يمكن استخدامه في جميع الصفحات

السطر التالي لتحديد مسار مجلد صفحات العرض و يمكنك أن تسميه ما تشاء .

و السطر الأخير لتحميل صفحة العرض المناسبة .

الآن ننشئ الملف **controller_base.php** في مجلد **clasess** و الكائن هذا سيكون هو الأب لجميع كائنات صفحات العرض التي قد تنشئها لاحقا

اكتب الآن الكود التالي :

```
<?php
Abstract Class Controller_Base {
protected $registry;
function __construct($registry) {
$this->registry = $registry;
}
abstract function index();
}

?>
```

في البداية يقوم بعمل نسخة عن كائن التسجيل حتى لا نقوم بنسخها في كل صفحة عرض و إنما ستعمم عن طريق الوراثة و قمنا بجعله **abstract** و كذلك أنشأنا الوسيلة **index** و جعلناها **abstract** و فائدة هذه الوسيلة **abstract** هو لضمان احتواء جميع كائنات صفحات العرض على وسيلة **index** .

لننشئ الآن أول صفحة عرض , قم بإنشاء الملف **index.php** في مجلد **controllers** و اكتب التالي :

```
<?php
Class Controller_Index Extends Controller_Base {
function index() {
echo '<font color=99000>محمد ديوب يرحب بكم</font> ';
}

}?>
```

لقد قمنا بإنشاء أول كائن عرض , لاحظ في السطر الأول قمنا بكتابة **Index_** بعد **Controller** وهو اسم عنصر التحكم الذي يجب أن يكون نفسه اسم ملف صفحة العرض و من ثم أنشأنا الوسيلة index

استعرض الآن النظام ستجده يعمل بكفاءة و لأي مشاكل أنا جاهز للمساعدة

خاتمة ...

كان هذا نظام بسيط جداً جداً و لكن الفائدة منه هو فهم طريقة عمل هذه الأنظمة ، أتمنى أن أكون قد أفدت و هي الغاية من هذا الكتيب و السلام عليكم و رحمة الله و بركاته

محمد ديوب

تم إنشاء هذه الوثيقة باستخدام برنامج OpenOffice.org Writer المفتوح المصدر و تم تحويله إلى ملف pdf بواسطة البرنامج أيضاً .

المراجع :

درس أجنبي مترجم و بعض صفحات المنتديات من هنا و هناك طبعاً مع بعض التصرف و التوسع في الشرح ...